DAIGL1

- Chapitre 4-

Les tableaux et les structures

A. Les tableaux

A.1. Introduction

Imaginons que dans un programme, nous ayons besoin simultanément de 12 valeurs (par exemple, des notes pour calculer une moyenne). Évidemment, la seule solution dont nous disposons à l'heure actuelle consiste à déclarer quinze variables, appelées par exemple Notea, Noteb, Notec, etc. Bien sûr, on peut opter pour une notation un peu simplifiée, par exemple N1, N2, N3, etc. Mais cela ne change pas fondamentalement notre problème, car arrivé au calcul, cela donnera obligatoirement une atrocité du genre :
Moy = (N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10 + N11 + N12) / 12

Autre exemple :

Saisir la liste des 12 notes sur 30
16 23 8 19 28 20 18 14 10 9 15 24
Voici la liste de ces notes sur 20
10.67 15.33 5.33 12.67 18.67 13.33 12 9.33 6.67 6 10 16

Dans l'état actuel de vos connaissances, pour écrire l'algorithme qui donne la sortie d'écran suivante, vous êtes obligé de déclarer 12 variables différentes. On ne peut pas utiliser une seule variable qui prend successivement chaque valeur. Passe encore avec 12 notes, mais si l'on voulait réaliser ce traitement avec 30 ou 40 notes, cela deviendrait fastidieux.
En outre, le même traitement est effectué 12 fois sur des variables différentes. Comme les variables ont des noms différents, on ne peut pas utiliser de boucle, ce qui allonge considérablement le code et le rend très répétitif.
Pour résoudre ces problèmes, il est tentant d'utiliser un nom commun pour toutes les variables et de les repérer par un numéro. Ainsi, on pourrait déclarer toutes les variables d'un seul coup et utiliser une boucle pour effectuer le traitement en faisant varier le numéro des variables. Cela est possible grâce à l'utilisation d'un tableau.

```
Algorithme conv_note
Variables
note : tableau[1..12] de réels
i : entier
Début
    Afficher "Saisir la liste des 12 notes sur 30"
    Pour i ⇦1 jusqu'à 12 Faire
        Saisir note[i]
    FinPour
    Afficher "Voici la liste de ces notes sur 20"
    Pour i ⇦ 1 jusqu'à 12 Faire
        Afficher note[i]*2/3
    FinPour
Fin
```

Un tableau est un ensemble de plusieurs composants de même type regroupés sous un nom collectif. On le représente souvent comme une suite de **cases** contenant chacune une valeur.

Ex : Tableau salaire

1ère case	2ème case	3ème case	4ème case	…	19ème case	20ème case
1524	1296	1875	1982	…	991	1494

Un tableau possède un **nom** (ici salaire) et un nombre d'**éléments** (de cases) qui représente sa **taille** (ici 20).

Tous les éléments d'un tableau ont le même type.

Pour désigner un élément, on indique le nom du tableau suivi de son **indice** (son numéro) entre crochets (ou entre parenthèses) :

salaire [3] représente le 3ème élément du tableau salaire et vaut 1875 euros.

La notion de "case contenant une valeur" doit faire penser à celle de variable. Et, en effet, **les cases du tableau**, encore appelées éléments du tableau, sont des **variables**, qualifiées d'**indicées**.
- les **variables classiques** sont déclarées individuellement et ont un nom distinct ;
- les **variables indicées** (constituant le tableau) sont **implicitement déclarées** lors de la déclaration du tableau. Pour bien montrer que ces variables n'ont pas de signification propre, mais sont juste "une des variables du tableau", elles ne possèdent pas de nom propre, mais juste un numéro appelé indice (de 1 à *n* si le tableau possède *n* cases). Chaque case (variable) est donc totalement identifiée par son indice et le nom du tableau.

Le tableau lui-même constitue aussi une variable. C'est une **variable complexe** (par opposition aux variables simples) car il est constitué d'autres variables (les éléments du tableau).

A.2. Déclaration et manipulation des tableaux à une dimension

A.2.1. Déclaration

La syntaxe de la déclaration d'une variable tableau est la suivante :

Nom du tableau **: tableau** *[valeur_indice_minimum* **..** *valeur_indice_maximu***m] de** *type_des_éléments*
Ex : sal : tableau[1.. 20] de réels
notes : tableau [1..10] de réels
nom_clients: tableau [1..20] de chaînes

Remarques :
Lorsqu'on déclare un tableau, on déclare aussi de façon implicite toutes les variables indicées qui le constituent. **En règle générale, l'indice minimum vaut 1**. Mais on peut aussi utiliser un autre indice minimum, comme 0. Dans ce cas, l'indice maximum sera égal au nombre d'éléments – 1. En langage C l'indice minimum est de 0.

Il est parfois utile de déclarer un tableau constant, c'est-à-dire un tableau dont les valeurs ne peuvent pas changer. Dans ce cas, comme pour toutes les autres constantes, il faut initialiser les valeurs du tableau dès la déclaration. Pour cela, on utilise une liste d'initialisation entre accolades :

Exemple : Voici un tableau constant qui mémorise le libellé des mois :
libmois[1..12] ={"janvier", "février", "mars", "avril", "mai", "juin", "juillet", "août", "septembre", "octobre", "novembre", "décembre"}

A.2.2. Manipulation

Les éléments d'un tableau sont des variables indicées qui **s'utilisent exactement comme n'importe quelles autres variables classique**s. Autrement dit, elles peuvent faire l'objet d'une affectation, elles peuvent figurer dans une expression arithmétique, dans une comparaison, elles peuvent être affichées et saisies…
L'**indice** d'un élément peut être :
- directement une valeur ex : salaire [10] ;
- une variable ex : salaire [i] ;
- une expression entière ex : salaire [k+1] avec k de type entier.

Quelque soit sa forme, **la valeur de l'indice** doit être :
- **entière ;**
- **comprise entre les valeurs minimales et maximales** déterminées à la déclaration du tableau.

Par exemple, avec le tableau tab [1..20], il est impossible d'écrire tab[0] et tab[21]. Ces expressions font référence à des éléments qui n'existent pas.

Le fait que les éléments constituant un tableau soient indicés permet de **parcourir tous les éléments avec une boucle**. Le ***Pour*** est généralement la boucle la plus adaptée, puisque l'on connaît le nombre de fois qu'on doit effectuer le traitement (une fois par élément). On utilise une variable qui sert d'indice et s'incrémente à chaque tour de boucle.

Application 1 :
Effectuer la trace pour chacun des algorithmes suivants :

```
Algorithme numéro1
Variables
Tab : Tableau[1..4] de Entier
Début
      Tab(4) ⇦ 4
      Tab(2) ⇦ Tab(4) + 2
      Tab(1) ⇦ Tab(2) + Tab(4)
      Tab(3) ⇦ Tab(4) / Tab(1)
Fin
```

```
Algorithme numéro2
Variables
T : Tableau[1..6] de Entier
I : Entier
Début
      Pour I ⇦ 1 à 6 Faire
           T(I) ⇦ T(I) + I
      FinPour
Fin
```

```
Algorithme numéro3
Variables
T : Tableau[1..6] de Entier
I, X, K : Entier
Début
        …
        K ⇦ 6
        Pour I ⇦ 1 à 6 Faire
                X ⇦ T(K)
                T(K) ⇦ T(I)
                T(I) ⇦ X
                K ⇦ K – 1
        FinPour
Fin
```

T	5	1	8	22	2	3

```
Algorithme numéro4
Variables
NOM : Tableau[1..5] de Chaîne de caractères
SUIV : Tableau[1..5] de Entier
TETE, Adr : Entier
Début
        …
        Adr ⇦ TETE
        TanT Que Adr <> 0 Faire
                Afficher NOM(Adr)
                Adr ⇦ SUIV(Adr)
        FinTQ
Fin
```

NOM	PAUL	ANDRE	HENRI	ANNE	LEA

TETE	2

SUIV	0	4	5	3	1

B. Les types structurés de tableaux

B.1. Introduction

Contrairement aux tableaux qui sont des structures de données dont tous les éléments sont de même type, les enregistrements sont des structures de données dont **les éléments peuvent être de type différent** et qui se rapportent à la **même entité** (cf. Merise).

Les éléments qui composent un enregistrement sont appelés **champs**. Avant de déclarer une variable enregistrement, il faut avoir au préalable définit son type, c'est à dire le nom et le type des champs qui le compose. Le type d'un enregistrement est appelé **type structuré**. (Les enregistrements sont parfois appelé structures, en analogie avec le langage C).

B.2. Déclaration d'un type structuré

Jusqu'à présent, nous n'avons utilisé que des types primitifs (caractères, entiers, réels, chaînes) et des tableaux de types primitifs. Mais nous pouvons créer nos propres types puis déclarer des variables ou des tableaux d'éléments de ce type.
Pour créer des enregistrements, il faut déclarer un nouveau type, basé sur d'autres types existants, qu'on appelle type structuré Après avoir défini un type structuré, on peut l'utiliser comme un type normal en déclarant une ou plusieurs variables de ce type. Les variables de type structuré sont appelées enregistrements.

La déclaration des types structurés se fait dans une section spéciale des algorithmes appelée Type, qui précède la section des variables (et succède à la section des constantes).

Syntaxes :

(notation inspirée du Pascal)	(notation inspirée du C)
Type *nom_type* = enregistrement *nom_champ*1 : *type_champ1* … *nom_champ*n : *type_champn* Finenreg	Type Structure *nom_type* *nom_champ1 : type_champ1* … *nom_champN : type_champN* FinStruct
Exemple : Type tpersonne = enregistrement nom : chaîne prénom : chaine âge : entier Finenreg	Exemple : Type Structure tpersonne nom : chaîne prénom : chaîne âge : entier FinStruct

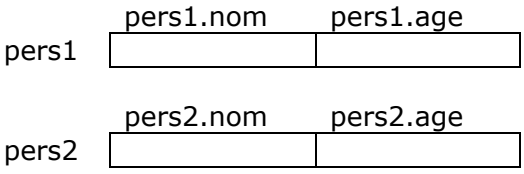
B.3. Déclaration d'un enregistrement à partir d'un type structuré

Une fois qu'on a défini un type structuré, on peut déclarer des variables enregistrements exactement de la même façon que l'on déclare des variables d'un type primitif.

Syntaxe :
Variables
nom_var : *nom_type*

Exemple :
Variables
pers1, pers2, pers3 : tpersonne

Représentation : les enregistrements sont composés de plusieurs zones de données, correspondant aux champs :



B.4. Manipulation d'un enregistrement

La manipulation d'un enregistrement se fait au travers de ses champs. Comme pour les tableaux, il n'est pas possible de manipuler un enregistrement globalement, sauf pour affecter un enregistrement à un autre de même type. Par exemple, pour afficher un enregistrement il faut afficher tous ses champs uns par uns.

B.4.1. Accès aux champs d'un enregistrement

Alors que les éléments d'un tableau sont accessibles au travers de leur indice, **les champs d'un enregistrement sont accessibles à travers leur nom, grâce à l'opérateur '.' :**

nom_enregistrement . nom_champ

représente la valeur mémorisée dans le champ de l'enregistrement

Par exemple, pour accéder à l'âge de la variable pers2, on utilise l'expression :

pers2.âge

Attention : le nom d'un champ est TOUJOURS précédé du nom de l'enregistrement auquel il appartient. On ne peut pas trouver un nom de champ tout seul, sans indication de l'enregistrement.
Les champs d'un enregistrement, tout comme les éléments d'un tableau, sont des variables à qui on peut faire subir les mêmes opérations (affectation, saisie, affichage,…).

Exemple 1 :
Programme de saisie des données concernant les personnes pers1 et pers2, puis affichage de la différence d'âge entre ces deux personnes :

```
Algorithme Exemple
Type
Structure tpersonne
        nom : chaîne
        prénom : chaîne
        âge : entier
FinStruct
Variables
pers1, pers2 : tpersonne
Début
        Afficher "Entrez le nom puis l'age de la personne 1"
        Saisir pers1.nom, pers1.age // il est impossible d'écrire Saisir pers1
        Afficher "Entrez le nom puis l'âge de la personne 2"
        Saisir pers2.nom, pers2.age
        Afficher "La différence d'âge entre ", pers1.nom, " et ", pers2.nom, " est de "
        Si pers1.age > pers2.age Alors
                Afficher pers1.age – pers2.age, " ans "
        Sinon
                Afficher pers2.age – pers1.age, " ans "
        FinSi
Fin
```

Passage d'un enregistrement en paramètre d'un sous-programme (cf. chapitre suivant)

Il est possible de **passer tout un enregistrement en paramètre** d'une fonction ou d'une procédure (on n'est pas obligé de passer tous les champs uns à uns, ce qui permet de diminuer le nombre de paramètres à passer), exactement comme pour les tableaux.

Exemple :
Voilà une fonction qui renvoie la différence d'age entre deux personnes :

```
Fonction différence (p1, p2 : tpersonne)
Début
        Si pers1.age > pers2.age ALors
                Retourne ( pers1.age – pers2.age )
        Sinon
                Retourne ( pers2.age – pers1.age )
        FinSi
FinFonct
```

Exemple 2 :
Voilà une procédure qui permet de modifier le prix de vente hors taxes d'un produit passé en paramètre. Cette procédure commence par afficher le libellé et l'ancien prix de vente hors taxes du produit puis saisit le nouveau prix de vente entré par l'utilisateur.

```
Procédure majpv (E/S x : produit)
Début
        Afficher "produit: ", x.lib
        Afficher "prix de vente hors taxe actuel: ", x.pvht
        Afficher "Entrez le nouveau prix de vente: "
        Saisir x.pvht
        Afficher "le nouveau prix de vente est: ", x.pvht
FinProc
```

B.4.2. L'imbrication d'enregistrements

Supposons que dans le type personne, nous ne voulions plus l'âge de la personne, mais sa date de naissance. Une date est composée de trois variables (jour, mois, année) indissociables. Une date correspond donc à une entité du monde réel qu'on doit représenter par un type enregistrement à 3 champs.
Si on déclare le type date au préalable, on peut l'utiliser dans la déclaration du type personne pour le type de la date de naissance.

Un type structuré peut être utilisé comme type pour des champs d'un autre type structuré :

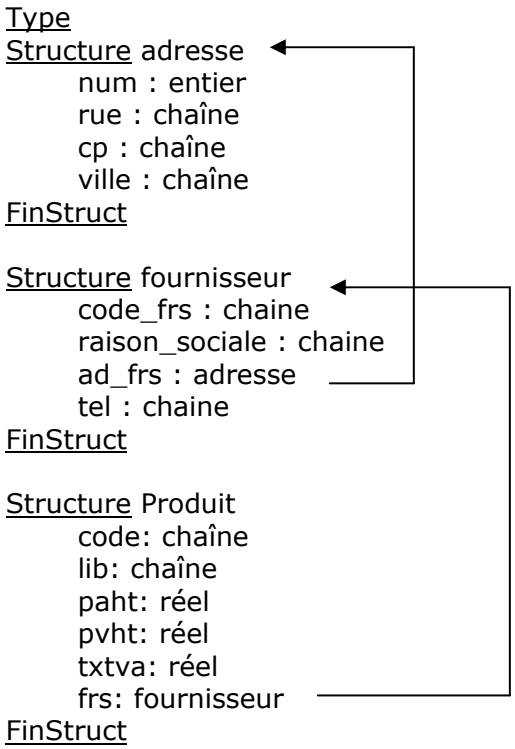
```
Type
Structure date
        jour : entier
        mois : chaîne
        année : entier
FinStuct

Structure personne
        nom : chaîne
        ddn : date
FinStruct
```

Pour accéder à l'année de naissance d'une personne, il faut utiliser deux fois l'opérateur '.' :
pers1.ddn.année

Il faut lire une telle variable **de droite à gauche :** l'année de la date de naissance de la personne 1.

Un produit est livré par un seul fournisseur. Un fournisseur est caractérisé par son code, sa raison sociale et son numéro de téléphone.



```
Type
Structure adresse
        num : entier
        rue : chaîne
        cp : chaîne
        ville : chaîne
FinStruct

Structure fournisseur
        code_frs : chaine
        raison_sociale : chaine
        ad_frs : adresse
        tel : chaine
FinStruct

Structure Produit
        code: chaîne
        lib: chaîne
        paht: réel
        pvht: réel
        txtva: réel
        frs: fournisseur
FinStruct

Variables
P : produit
```

Voilà l'instruction qui permet d'afficher le numéro de téléphone du fournisseur du produit p.frs.tel :
Afficher "téléphone du fournisseur de ", p.lib, " : ", p.frs.tel

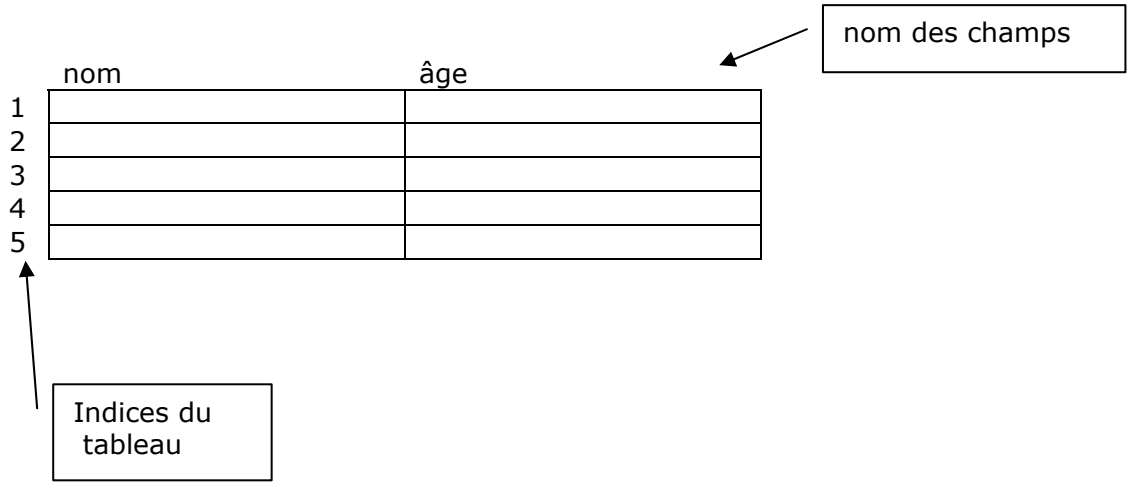
B.5. Les tableaux d'enregistrement (ou tables)

Il arrive souvent que l'on veuille traiter non pas un seul enregistrement mais plusieurs. Par exemple, on veut pouvoir traiter un groupe de personne. On ne va donc pas créer autant de variables du type personne qu'il y a de personnes. On va créer un tableau regroupant toutes les personnes du groupe. Il s'agit alors d'un tableau d'enregistrements.

```
Constantes
NP = 20 // nombre de personnes du groupe
Type
Structure personne
        nom: chaîne
        age: entier
FinStruct
Variables
Groupe : tableau[1..NP] de personnes
…
```

Chaque élément du tableau est un enregistrement, contenant plusieurs variables de type différent. On accède à un enregistrement par son indice dans le tableau.

groupe[2] représente la deuxième personne du groupe ;
groupe[2].nom représente le nom de la deuxième personne du groupe.



Attention : groupe.nom[3] n'est pas valide. Pour accéder au nom de la troisième personne du tableau, il faut écrire : groupe[3].nom

Olivier Mondet
http://unidentified-one.net